In the Specification:

The following insertions are made with citations to the specification as published as US 2004/0167880 A1. The insertions are indicated by underscoring.

Please replace paragraph **[0030]** with new paragraph **[0030]** shown below.

**[0030]** A virtual or federated content repository (hereinafter referred to as "VCR") **100** is a logical representation of one or more individual content repositories **108** such that they appear and behave as a single content repository from an application program's standpoint. This is accomplished in part by use of an API (application program interface) **104** and an SPI (service provider interface) **102**. An API describes how an application program, library or process can interface with some program logic or functionality. The API 104 interface with the Applications 110. By way of a non-limiting illustration, a process can include a thread, a server, a servlet, a portlet, a distributed object, a web browser, or a lightweight process. An SPI describes how a service provider (e.g., a content repository) can be integrated into a system of some kind. SPI's are typically specified as a collection of classes/interfaces, data structures and functions that work together to ~~provided~~ provide a programmatic means through which a service can be accessed and utilized. By way of a non-limiting example, APIs and SPIs can be specified in an object-oriented programming language, such as JAVA™ (available from Sun Microsystems, Inc. of Mountain View, Calif.) and C# (available from Microsoft Corp. of Redmond, Wash.). The API and SPI can be exposed in a number of ways, including but not limited to static libraries, dynamic link libraries, distributed objects, servers, class/interface instances, etc.

Please replace paragraph **[0036]** with new paragraph **[0036]** shown below.

**[0036]** **Fig.3** is an illustration of objects used in connecting a repository to a VCR in one embodiment of the invention. In one embodiment, objects implementing API

interface RepositoryManager **302** can serve as ~~an~~ a representation of a VCR from an application program's standpoint. Repository Manager 302 connects to Application 300. A Repository Manager connect( ) method attempts to connect all available repositories with a current user's credentials to the VCR. By way of a non-limiting example, credentials in one embodiment can be based on the JAVA™ Authentication and Authorization Service (available from Sun Microsystems, Inc.). Those of skill in the art will recognize that many authorization schemes are possible without departing from the scope and spirit of the present embodiment. ~~Each available content repository is represented by an SPI Repository object **306-310**.~~ Each available content repository **312-316** is represented by an SPI Repository object **306-310**. The RepositoryManager object invokes a connect( ) method on a set of Repository objects. In one embodiment, a RepositorySession object (not shown) can ban be instantiated for each content repository to which a connection is attempted. In one embodiment, the RepositoryManager connect( )method can return an array of the RepositorySessions to the application program, one for each repository for which a connection was attempted. Any error in the connection procedure can be described by the RepositorySession object's state. In another embodiment, the RepositoryManager connect ( ) method can connect to a specific repository using a current user's credentials and a given repository name. In one embodiment, the name of a repository can be a URI (uniform resource identifier).

Please replace paragraph **[0048]** with new paragraph **[0048]** shown below.

**[0048]** **Fig. 8** is an illustration of a content editor **800** in one embodiment of the invention. Navigation pane **802** is in "content" mode **812** such that it selectively filters out nodes that define only schemas. Content node **806** ("Laptop") has been selected. Node **806** is a child of hierarchy node "Products", which itself is a child of repository "BEA Repository". Selection of node **806** causes a corresponding content node editor to be rendered in editor window **804**. The editor displays the current values for the selected node. The content type **814** indicates that the schema for this node is named "product". In this example, the node has five properties: "Style", "Description", "Color", "SKU" and

"Image". A user is allowed to change the value associated with these properties and update the VCR (via the update button **808**), or remove the node from the VCR (via the remove button **810**).

Please replace paragraph **[0049]** with new paragraph **[0049]** shown below.

**[0049]** **Fig 9** is an illustration of a schema editor **900** in one embodiment of the invention. Navigation pane **902** is in "type" mode **910** such that it only displays nodes that have schemas but no content. Schema node **906** ("product") has been selected. Node **906** is a child of repository "BEA Repository". Selection of node **906** causes a corresponding schema editor to be rendered in editor window **904**. The editor displays the current schema for the selected node (e.g. derived from ObjectClass, Property Definition, Property Choice objects). In this example, the node has five property definitions: "Style", "Description", "Color", "SKU" and "Image". For each property, the editor displays an indication of whether it is the primary property, its data type, its default value, and whether it is required. A property can be removed from a schema by selecting the property's delete button **912.** A property can be added by selecting the "add property" button **908.** A property's attributes can be changed by selecting its name **914** in the editor window or the navigation pane **906** (see **FIG.10**).

Please replace paragraph **[0050]** with new paragraph **[0050]** shown below.

**[0050]** **Fig. 10** is an illustration of a property editor **1000** in one embodiment of the invention. The schema named "product" is being edited. Schema properties definitions are listed beneath their schema name in the navigation pane **1002**. Schema property **1008** ("color") has been selected. The editor window **1004** displays the property's current attributes. The name of the attribute (e.g., "color"), whether the attribute is required nor not, whether it is read-only, whether it is the primary property, its data type, default

value(s), and whether the property is single/multiple restricted/unrestricted can be modified. Changes to the a property's attributes can be saved by selecting the update button **1006**.